UNCLASSIFIED

10 October 1979

DBMS measurements

ON FILE, DOC RELEASE INSTRUCTIONS APPLY

MEMORANDUM FOR THE RECORD

SUBJECT: Visit with Elizabeth Fong, National Bureau of Standards

STATINTL 1. On 10 October 1979, [redacted] visited Mrs. Elizabeth Fong of the Network Architecture Group, Institute of Information and Computer Sciences, National Bureau of Standards, Gaithersburg, Maryland. The objective of the visit was to identify methodology for measuring data base management system resource demands from computer complexes. She had done some work on internal contract to the FAA in late 1974 and early 1975 dealing with the DBMS/Resource Utilization issue from an experimantal viewpoint.

2. The result of the visit with respect to the objective was that no such resource demand measure currently exists in her experience. She gave me a copy of the paper that she published on the subject. It is Attachment 1. She suggested contact with FEDSIM, Dr. Dennis Conti at NBS, and with De Lutis (late of Ohio State University, now with his own firm here in Washington working with the Social Security Administration).

3. Her current interest is a pilot system for access to a heterogeneous collection of Data Bases available through heterogeneous DBMS on hetereogeneous mainframes connected to a network. It is the pardigmatic Distributed Data Base. Her current paper is attached- Attachment 2.

Attachments:
As stated

[redacted]

STATI

UNCLASSIFIED

ATTACHMENT 1

IEEE Catalog No. 75 CH0988-6C

COMPCON

September 9-11 Fall

Eleventh IEEE Computer Society Conference
Mayflower Hotel, Washington DC

A BENCHMARK TEST APPROACH FOR GENERALIZED DATA BASE SOFTWARE*

Elizabeth Fong
Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

ABSTRACT

A benchmark test approach for generalized data base software is described. Although the benchmark test has been designed to experiment on one specific data base management system, the approach is rather general and applicable to many data base management systems that are currently available. The benchmark test consists of the specification of a test data base and the specification of a set of processing transactions to exercise the candidate software on the test data base. This benchmark test can be used both to measure the performance of a data base management system, and as a saturation test. The two parameters to be used in the saturation test are data base size and workload. The data base size is varied by increasing the number of records within the test data base. The workload is varied analytically via a queueing model.

INTRODUCTION

The work described in this paper is part of a project to design and develop a benchmark test for generalized Data Base Management Software (DBMS). Generalized DBMS in our context is the class of systems that provides for:

- operations on data such as data definition, data storage, data maintenance, data retrieval and data output,
- reference to data by name and not by physical location,
- an environment that is independent of a particular set of application programs or files.

These DBMS systems provide a single flexible facility for accommodating different data files and operations while demanding less programming effort than a conventional programming language, e.g., COBOL. These advantages are gained with potential risk of excessive software overhead and concomitant unsatisfactory performance. Substantial funds are repeatedly devoted to comparative analyses of DBMS. However, the attempt to determine whether candidate software has acceptable functional quality and performance cannot be definitively answered. In particular, many managers are concerned with the lack of methodology for determining whether present software will hold up under the stress of expanding data bases and inquiry workloads. A simple, benchmark test package that is relatively general, portable across DBMS systems, and representative of data base functions and applications is strongly needed.

This paper described the design of such a benchmark test package. A specific implementation of this general benchmark test approach is being performed for a selected DBMS.

The primary objective of this tool is to provide uniformity in data base testing. In particular, system tests are constructed which demonstrate the effectiveness of a DBMS system with growing file sizes and processing loads. The method developed can be used for comparing the performance of different data base management software as an aid in system selection.

THE APPROACH

In scientific computation, system loads for benchmark tests are typically specified as instruction mixes, procedural program mixes, etc. However, these are not quite appropriate for the benchmarking of DBMS. Also, the typical analytical studies of file organization performance tend to simulate essentially single record accesses and do not bear relevance to real life DBMS operations.

The principal components of this benchmark test approach involve the development of a transferable test data base and a set of process transactions to exercise the candidate software. The transactions are primitive data handling functions such as retrieval with conditions, new data entry, etc. For the test on the selected system, eight of these functions are defined. The data elements and their values required for the test transactions are carefully chosen so that the results are known.

The test environment is kept as simple as possible. The DBMS and its hardware interface are assumed to be typical of an intended installation. Since this benchmark test involves on-line timing measurements, the benchmark runs are conducted on the system with the communication system intact.

The parameters to be varied are:

- performance of DBMS with respect to growing data base size;

*This work has been supported by the Federal Aviation Administration under Interagency Agreement No. DOT-FA74WAI-474.

performance of DBMS with respect to increasing workloads.

Data base size is varied by changing the number of records within the test data base. Individual test transactions are executed and the timing measurement taken. Then the data base size is changed and the same transactions are executed again. The timing measurement is recorded in each size iteration.

The increasing workload parameters are obtained analytically, since it is difficult to achieve a multi-terminal concurrency environment without having a software package such as a terminal environment simulator. The test provides an empirical measurement of the individual response time for a single terminal input. These measurements are inputs to a queueing model. The empirical data are extrapolated by the model to predict how response time will vary as the number of terminals is increased.

The overall response time for a given process (call it T_R) is expressed as $T_R = T_C + T_P + W$, where T_C is the communications delay, T_P is the processing time and W is the estimated wait time. The W, which is the waiting time in the queue, is obtained through an analytic model as a function of increasing number of active terminals.

GENERATION OF A TEST DATA BASE

In order to test a particular DBMS, a data base must be constructed for it. The basic building block in the construction of a test data base is the attribute of the data element. The collection of data element attributes forms the structure of a record. Only the logical record structure needs to be considered, because it is defined through the candidate DBMS using the data definition language.

Instead of generating a totally fictitious data base, a "live" data base is used with a subset of the data elements identified. These data elements, with known values randomly distributed within the "live" data base, participate in the test transactions. The rest of the data elements are not referenced in the test transactions.

The data elements are identified by their attributes. The attributes are various combinations of the following individual data characteristics: type, length, key, unique, required, linked, grouped, and optionally hierarchical level. The total combination of these data characteristics makes up the "data element attribute".

The test data base is the existing data base plus those specific test data element attributes identified. These test data element attributes will have predetermined data content. The data content of the remaining fields in the records and all other non-test records will not be directly referenced in the benchmark testing.

The number of records within the main file is controlled. For the selected system implementation, the numbers of records in the test data base are 25,000, 50,000, 100,000, 200,000 and 500,000.

Construction of the test data base involves the following steps:

1. Define the attribute names discussed above. These must, of course, be in the class of data characteristics supported by the candidate DBMS.
2. Define the data elements in the data definition language of the candidate DBMS.
3. Generate test data and merge it with an existing data base. If an existing data base is not used, write a special computer program to generate values for the records. Test data generators are commercially available for that purpose.
4. Load the test data base for use by the candidate DBMS.

The preparation of the data base is the major component of the benchmark test process. The crucial part of this test data base is the identifiable logical data element attributes. It is these data element attributes with assigned values, that participate in the execution of test transactions. The advantages of this test data base are that an existing data base can be used with a subset containing those identifiable logical data element attributes.

TEST TRANSACTIONS

The benchmark programs represent the primitive functions of any data base management system. The procedure specifications expressed in "near-English" style are translated into specific DBMS self-contained language. For the specific selected DBMS experiment, eight test transactions are defined. The amount of checking and the type of error messages are all specified. To avoid lengthy detail specification here, only the functional descriptions of the eight test transactions are given below:

TEST 1 - On-line data entry function. This test transinvolves (a) the display of a screen template without data values, (b) the filling of the screen by an operator, (c) the validation of entered data, and (d) the placing of the validated data into a record of the main file so that it can be retrieved.

TEST 2 - On-line data update function involving modification of an existing data element with a new value of the same length and type.

TEST 3 - On-line data update function involving deletion of a whole record in the secondary file and associated

data element entries in the main file.

TEST 4 - On-line data retrieval involving counting the number of occurrences of a named data element having a specified value. A named data element having a comparative relationship such as equal to, greater than etc. with a specified value is called a triplet. The data element attribute must possess the following characteristics: type = either alphanumeric or numeric, length = don't care, key = yes, unique = no, required = yes, linked = don't care, grouped = no, and hierarchical level = first level.

TEST 5 - On-line data retrieval with at least two triplets combined with and "AND" logical connective, and a listing of at least four data elements. The data element attribute for the first triplet and the second are specified as in TEST 4.

TEST 6 - On-line data retrieval involving two files (i.e., one triplet is to be defined in another file) and a printout of at least four data elements in the first file. Both data element attributes are indicated as in TEST 4.

TEST 7 - On-line retrieval command entry but deferred for off-line printing. For this test, timing is initiated when the request is entered on the terminal. Timing is terminated when the output is formatted ready to print, but not actually printed out. The output should contain a title, a column heading and variations on single and double spacings.

TEST 8 - A request containing a retrieval function with an arithmetic add function to be predefined with at least three parameters. Catalog this request and invoke it with actual parameters. Timing should start when the predefined request is being invoked by actual parameters.

A data base of proper size must be defined, created, and loaded prior to running these benchmark test transactions. The test transactions must be debugged to assure runability. To insure the validity of the results, the computer configurations must remain constant between tests.

For the selected DBMS experiment, four sets of input data values are defined to be used for the execution of these eight test transactions. The mean value of the test results is then used to discount the effect of record positions on the mass storage device.

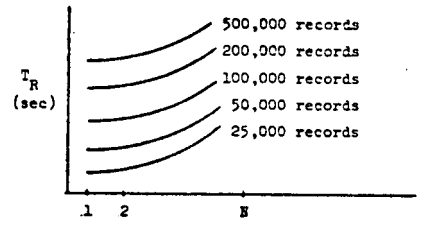
PROJECTION OF TEST RESULTS

The response time measured is the individual response time to execute an individual test transaction from a single terminal with no other jobs in the computer. Response time is measured from the point at which the request is entered on the screen to the point at which the response is received on the screen. The operator's time for typing in the requests is excluded. Processing time and the communications delay is included.

For the selected DBMS experiment, the timing measurements of these individual test transactions are recorded on a specially designed form. The measurements taken are as follows: for a fixed file size, each of the eight test transactions is executed individually with four different sets of parameter values to obtain an average response time. These response times are then combined to obtain a mean response time T_R.

T_R is the total overall response time for a fixed file size, with one terminal. For the next iteration, the file size is increased. The data is reinitialized, and the eight test transactions are executed again with the same sets of parameter values. Thus another T_R for a larger file size is obtained. This is repeated five times for files of 25,000, 50,000, 100,000, 200,000 and 500,000 records.

Finally a single queue, single server, first-come, first-serve model is used to extrapolate response times for increasing number of terminals. This model description is the subject of another paper. Ultimately, the benchmark results project performance guidelines in the form of engineering graphs as suggested by the following figure:



SUMMARY

Interest in developing a uniform benchmark test methodology for generalized data base software has grown substantially, as a result of the proliferation of DBMS and the complex problem of evaluation and selection for a given application. This paper describes an approach for uniformly benchmark testing a DBMS.

The benchmark test package consists of a carefully controlled quality sample data base plus a set of processing transactions representing the range of processing requirements in user application environment. Performance comparisons are made by executing these transactions on various commercial systems. The speed of execution of each transaction will provide a

unit measurement for a known file size for a specific system. The response times measured can be used in three ways: the measurement data can be evaluated to assess the relative strengths and weaknesses of each DBMS. For instance, system "A" is faster, in retrieval while system "B" is faster in updating. The second use is to compare the execution speed across various commercial systems along the common set of test conditions, e.g., same size data base, same test transaction specifications. The third use of the measured response time is as input to an analytical model which extrapolates system performance as workloads are increased. This will yield a saturation point for the candidate DBMS with respect to growth potential.

A case study using this approach is being implemented for a selected data base management system to prove feasibility. Results will be available in the future.

Clearly, this simple test package does not measure several important aspects of DBMS, e.g., resource utilization, adaptability to change, ease of use, etc. However, the existence of such a set of Government-wide test routines, not now available, will permit agencies to directly compare the relative strengths and weaknesses of commercial software, without redeveloping the needed tests.

ACKNOWLEDGEMENT

The design of this benchmark test approach is a team effort. In particular, numerous ideas originated from Dennis Fife of NBS. The author wishes to acknowledge Joseph Collica and David Gilsinn for their technical contribution on this project.

XNDM: AN EXPERIMENTAL NETWORK DATA MANAGER

Stephen R. Kimbleton
Pearl S.-C. Wang
and
Elizabeth N. Fong

National Bureau of Standards
Washington, D.C. 20234

ABSTRACT

Data base access is increasingly important in a networking environment. Two alternative approaches can be identified: i) implementation of distributed databases presenting the user with one logical database implemented across a collection of computers or, alternatively, ii) development of network data managers providing a uniform user and program viewpoint across heterogenous DBMSs. While the first approach is the most natural extension of the concept of an individual DBMS, its utilization imposes certain requirements including the necessity for converting existing DBMSs if their data is to be supported in the distributed environment. The second approach minimizes or eliminates conversion problems: however, it has not yet been shown feasible. This paper describes an ongoing research project concerned with establishing the feasibility, issues, alternatives, and a technical approach for supporting a network data manager. Although implementation has not been completed, the initial evidence is positive and suggests that network data managers may well prove either an acceptable alternative or useful intermediate stage to a distributed database.

1. INTRODUCTION

Computer networks support the sharing of remote programs and data. The gradual maturation of networking technology, as measured by the increasingly sophisticated protocols and applications being implemented [ARPAN 76], [INWG 77], has resulted in increasing demands for supporting remote access to data.

 Partial funding for the preparation of this paper was provided by the U.S. Air Force Rome Air Development Center (RADC) under Contract No. F 30602-77-0068. Certain commercial products are identified in this paper in order to adequately specify the procedures being described. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the material identified is necessarily the best for the purpose.

To appear in: Proceedings of the 4th Berkeley Workshop on Distributed Database Management

An individual user, interacting with a remote database management system (DBMS), issues queries and updates in the data manipulation language (DML) used by the system and receives data in response. Because of differences in: i) the data model used in constructing DBMS supported data structures, ii) the functionality provided by the software even if the underlying data models are the same, iii) data structure, e.g. data base semantic differences which are also likely even if the same underlying data model is employed, iv) DML differences, and v) computer system differences, the user wishing to access multiple remote databases is faced with a substantial learning burden.

This paper argues that this learning burden can be substantially offloaded from the user. Accomplishing this requires a network data manager providing a uniform user viewpoint across multiple remote heterogeneous DBMSs. The feasibility of this approach is being explored through constructing an Experimental Network Data Manager (XNDM) at the National Bureau of Standards.

The basic assumption underlying the design of XNDM is heterogeneity of data models, data structures, DBMSs, DMLs and computer systems on which these DBMSs reside. Superimposing a uniform user viewpoint in such an environment clearly requires a substantial amount of software and may be a significant source of delay in processing user requests.

To explore this issue, recall that information processing requirements can be divided into three categories [ANTHR 65]: operational control, managerial control and strategic planning. As one passes from operational control to strategic planning, the bandwidth of the application decreases as does its predictability. Intuitively, we believe that network data managers are inappropriate for operational control, highly appropriate for strategic planning, and may be of help in managerial control. For example, handling inventory out-of-stock conditions could be simplified through a means for querying remote DBMSs to determine an alternative source of supply when an out-of-stock is indicated by the local DBMS.

The preceding suggests that strategic planning and exception reporting constitute two likely applications for a network data manager. Moreover, the nature of these applications suggests that the additional overhead of supporting a network data manager is likely to prove very acceptable in comparison with the burden of manually performing the necessary translation processes in response to unpredictable and non-recurrent demands.

The remainder of this paper provides a more detailed discussion of XNDM. To provide context, section 2 establishes some comparisons between a network data manager and a distributed database. Section 3 describes the user's view provided by XNDM. Section 4 discusses translation technology required to support this view and observes that it differs substantially from that currently discussed in the data translation literature. Section 5 describes the current XNDM implementation status and presents some concluding remarks.

2. NETWORK DATA SUPPORT OPTIONS

A distributed DBMS (DDBMS) is usually viewed as one logical DBMS implemented across several host computers. Thus, excluding performance differences, there is no apparent difference to the user in accessing a DDBMS and accessing a DBMS resident on a single host using the same data structures and data manipulation language. Moreover, through redundancy, the DDBMS potentially permits increased reliability and decreased access times to frequently used portions of the database. Redundancy does require care in ensuring consistency of multiple data copies and in synchronizing updates [ROTHJ 77], [STONM 77].

Using a DDBMS poses the need for conversion of existing DBMSs. The current state of database conversion suggests that non-trivial costs are associated with this process [NAVAS 76]. Moreover, even if these costs were insignificant, the resulting organizational dislocation in adapting to the new DDBMS is likely to be extensive. Consequently, the DDBMS approach may prove infeasible given the environment in which it is to be implemented.

A network data manager is intended to provide an alternative to the DDBMS through providing an easy means for simplifying network access to multiple, heterogeneous DBMSs. The basic relationship between a network data manager and the individual DBMSs is illustrated in Figure 2-1 in the context of the NBS Experimental Network Data Manager (XNDM). Thus, a process represented as a circle within one computer (PHOST) interacts in a uniform way with multiple independent DBMSs located in one or more computer systems.

Our working hypothesis is that the network data manager approach is likely to prove very acceptable in handling unpredictable and non-recurrent requests. Moreover, given the cost of database conversion, it is also likely to be the only feasible way of easily adapting to the opportunities for sharing information which are provided by networking. Thus, we are motivated to consider its design and development in greater detail.

3. THE NETWORK USER ENVIRONMENT

The two essential functions of a Network Data Manager are provision of a uniform user environment across individual (heterogeneous) local DBMSs (LDBMSs), and translating between this user environment and the LDBMSs. The remainder of this section structures the basic components of the XNDM supported user environment while the following section addresses translation technology.

3.1 Data Model/Data Language Selection

Developing a data language and data model for XNDM can be approached either as a problem of developing a 'best' data model and data language and then considering the issues in translating to existing data models and languages or through selecting one of the existing data models and languages. The former is a problem of independent interest. Requiring its solution as the prerequisite to analyzing network data managers seems undesirable. Instead, we have chosen to examine the existing alternatives, select a reasonable candidate, and place primary emphasis on the data manager specific aspects of the problem. This has expedited our consideration of the basic nature of the problem. It will be interesting to see if future data model/data language research can be easily accommodated as we expect or will, instead, require substantial revision.

Selection of a data model for XNDM has been driven by three basic assumptions. The first is that the network user is naive vis-a-vis the access requirements of local DBMSs. The second is that the network user should be assisted to ensure that queries and updates are meaningful. The third is that the local DBMS should be provided with relatively tight guarantees that the network user will not be able to adversely affect its operations through ignorance or intent. Note that the second and third assumptions are closely interrelated.

The first assumption motivates selection of a data model and data language minimizing the knowledge and effort required to support access. That is, the data model should present data in a way which is easy for the user to understand. Further, the Data Manipulation Language (DML) should minimize procedural (extent to which the user must specify how rather than what is to be retrieved or updated) and navigational (need for explicitly specifying interrelationships between data elements) requirements.

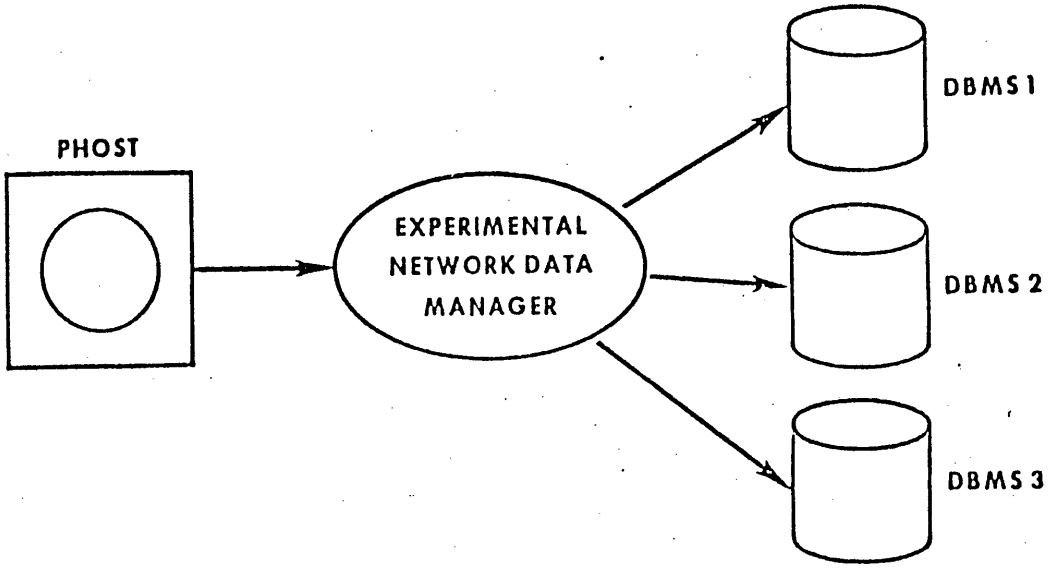


Figure 2-1. XNDM Interface between User Program and Multiple Remote DBMSs

Of the three basic data models: relational, hierarchical, and network, it is our opinion that the relational model is the simplest to understand. Accordingly, we have chosen tables as the basic mechanism for representing data. Although a properly chosen user schema can result in an appropriately simple user viewpoint regardless of the particular global schema employed, the static nature of such a schema conflicts with the random and unpredicatable nature of arriving requests.

Meeting the requirements of our second assumption is being accomplished through provision of a semantic integrity system to ensure meaningful queries and updates as discussed below. Moreover, an access control mechanism is also being implemented to ensure that the network user is only permitted to access data appropriate to his/her access rights. This is the basic tool for meeting the third requirement.

3.2 Global Schema Specification

Central to the specification of an XNDM global schema is the balancing of the conflicting requirements of the network users so as to provide a design that can satisfy the need of the "community" of users - as opposed to the need of any individual user.

As discussed above, a basic XNDM assumption is that a uniform user environment is to be superimposed on a highly heterogeneous collection of existing local DBMSs. This requires: i) a common view of data to be presented to the network user, and ii) a means for mapping from this common view to the target systems. Note that this common view need not contain all of the data in the local DBMSs. Rather, it will probably comprise only that data thought to be of common interest. This, in turn, is likely to be a subset of the data which local DBMS management is willing to make available to the network user. Since both of these selection processes are judgmental, we assume that the selection of data and its attributes is performed by a team (of database administrators?) responsible for the overall utilization of the network data manager.

Given this selection, and the resulting structuring using the described data model, the need arises for a suitable translation process. This translation process proves to be substantially different from that currently discussed in the data translation literature. It is discussed in some detail in the following section.

3.3 Experimental Network Data Language

The Experimental Network Data Language consists of three major components: i) Experimental Network Data Manipulation Language (XNDML), ii) Experimental Network Data Control Language (XNDCL), and iii) Experimental Network Data Definition Language (XNDDL).

Since the basic XNDM objective was exploring the feasibility of providing a uniform environment for the network user, we decided to adopt an existing DML and add any extensions which proved necessary. After some consideration, we have chosen SEQUEL [CHAMD 76] to provide the basic framework for XNDL since: i) it is a table based DML, and ii) it has been subjected to human factors oriented investigations which have improved the quality of its user interface [REISP 75].

Currently, the design of both the query and update portions of XNDML has been completed and implementation of the query portion is underway. Implementation of update capabilities is being deferred pending completion of the design of XNDCL and XNDDL.

Table 3-1 lists the six major categories of XNDML query commands. XNDML does not contain the SEQUEL sorting facilities and certain alternative ways of stating predicates. Sorting was eliminated because it adds little to demonstrating the feasibility of a network data manager and can be an expensive consumer of processing time on the host containing the LDBMS. XNDML is invoked via subroutine CALLs. Thus it does not have a host language interface corresponding to that provided by SEQUEL.

A major difference between XNDML and SEQUEL is the need to specify the target database. Three major alternatives can be identified: explicit specification, implicit specification, and specification of location as a virtual attribute.

The target database can be explicitly specified by using the statement DATABASE IS 'DATABASENAME'. The effect of this statement is to make all subsequent XNDML statements refer to this DATABASE until another target specification is encountered.

Implicit specification of the target database occurs when the user issues an XNDML statement without any target database specification or, equivalently, the specification DATABASE IS 'ALL'. In this case, a dictionary describing network accessible information is accessed to identify those databases containing information about the entities and relationships identified in the XNDML statement. The statement is then applied against each such database and the results aggregated.

The third and most sophisticated specification is through treatment of location as a virtual attribute. This permits one to construct queries in which the predicate applies to location as well as to entities and their attributes. Thus, assuming that the distance between sites is known, one can specify the site of the location to replenish an out-of-stock condition as a function of conditions prevailing at each relevant location. For instance, an out-of-stock replenishment rule might be to replenish in an amount inversely proportional to distance and directly proportional to stock on hand. Distance proportionality can be used to lower shipping cost overhead while stock on hand proportionality could be used to avoid unduly impacting a site with a low stock level.

TABLE 3-1. XNDML Query Categories.

C1 SELECT (columns)

C2 SELECT....WHERE (rows)

C3 PARTITION

C4 SET OPERATIONS

C5 AGGREGATION

C6 COMPOSITION

3.4 Semantic Integrity

Semantic integrity is a significant issue in the context of an individual DBMS since it provides a means of assuring that the database is a valid representation of the application environment. Two major reports [MCLED 76] and [BRODM 78] have appeared on this subject as well as a variety of papers. The general objective is ensuring that if one starts with a valid DBMS configuration, subsequent updates will not impair this validity.

Semantic integrity promises to be of even greater importance in the context of a network data manager since local DBMS management is likely to want strong assurances that remote, and therefore presumably less knowledgeable users, will not affect DBMS integrity. This problem varies somewhat from that for an individual DBMS since XNDM cannot assure that the database is, initially, in a consistent state. Thus, the major concern is that updates are semantically correct. A lesser concern is facilitating the correct structuring of queries through supporting strong domain typing.

XNDM semantic integrity concerns also differ from the corresponding problem for an individual DBMS because the network user's view of data is virtual. Thus, there is a premium on performing all non-data dependent integrity checking before proceeding with the data dependent checks. This may ultimately result in a partitioning of integrity checking functions between XNDM and the LDBMS. In any event, the major issues can be divided into two major categories: i) assurance of integrity at the network level, and ii) assurance of integrity at the local DBMS level.

Although work on the XNDM Semantic Integrity System is in its preliminary stages [FONGE 79], some initial observations can be made. Semantic integrity can be expressed at the global schema level through the (virtual) tabular data model. Assuring integrity within an individual table can be subdivided into assurance of attribute integrity, row integrity, column integrity, and predicate integrity.

Assurance of semantic integrity is provided via two facilities: strong domain typing and predicate-based assertions. Strong domain typing facilities of XNDM permit the user to define: i) the format of the data, ii) the acceptable range of values, iii) the collection of legal (arithmetic, logical and string) operations, and iv) the interrelationships among data elements in terms of the collection of legally acceptable operations.

Predicate-based assertions specify validity criteria which are to hold in the application environment. The facility provided in XNDM will permit: i) specification of rules for consistency and correctness of data bases, ii) the time at which the assertion is to be enforced, and iii) the actions to be taken when the assertions are not satisfied.

Assuring predicate-based integrity for either an individual relation or for a collection of relations can imply significant overhead depending on the amount of data involved and the types of checks which must be performed.

3.5 Access Controls

A second major support function required for acceptance of XNDM is provision of an appropriate access control mechanism. Currently, many DBMSs provide access controls via passwords on files [DATEC 77]. This is clearly insufficient for the level of functionality intended to be provided by XNDM. The issue is whether a significantly better system can be implemented. This issue has been discussed in [WOODH 79]; the following summary considerations are based on the discussion contained therein.

Access control mechanisms can be divided into two major categories [RATGP 77]: i) non-discretionary access control mechanisms which support organizational constraints on the sharing of information, and ii) discretionary access control mechanisms which permit user directed controlled sharing of information.

Security levels and compartments constitute a major example of non-discretionary access control mechanisms. Conceptually, a user is labelled with security level(s) and compartments, e.g. level is SECRET, compartment is NATO, and is entitled to access all information having the same, or lower levels, e.g. level is SECRET or CONFIDENTIAL, compartment is NATO.

System-R provides an example of a sophisticated DBMS discretionary access control mechanism [GRIFP 76]. Through its use, an individual user is permitted to grant a subset of his/her access rights to another user. The supported functionality permits READing, INSERTing, DELETEing, UPDATEing, and DROPing (of an entire table). Moreover, a GRANT command permits one user to provide another user with the ability to GRANT rights. These mechanisms are supported for both an entire table and for individual columns of a table.

XNDM provides both discretionary and non-discretionary access controls. Their combined support requires a mechanism for checking that discretionary grants do not conflict with non-discretionary controls. This checking process has been implemented using the lattice security model [DENND 76].

4. TRANSLATION TECHNOLOGY

This section: i) establishes the differences between data translation required to support XNDM and that currently considered in the data translation literature, ii) discusses the two major alternatives in implementing a translation capability, and iii) describes the translation process which we have selected. Currently, translation has only been implemented for the query portion of XNDML which, for simplicity, we refer to as the Experimental Network Query Language (XNQL).

4.1 The Nature of the Translation Process

Data translation can be taxonomized in two different dimensions: i) online vs. offline, and ii) constraints on source and target data structures. XNDM translation requirements differ from those usually discussed in the data translation literature since: i) it is a real-time, online process, and ii) it is dependent upon both source and target data structures.

The requirement that the translation process be real-time and online forces a substantially different translation process than that usually considered in the context of database translation [NAVAS 76]. Specifically, the need for explicit consideration of physical representations of data is eliminated while the need for an online and realtime level of functionality cannot be avoided.

XNDM translation also differs from that usually associated with database front ends and database terminals. (A database front end presents the user with data structures differing from those actually employed by the DBMS being accessed and often based on a different data model. Thus, there is substantial interest in relational front ends to DBTG DBMSs. For a front end, the data structures presented to the user are 'fixed' and the data structures employed by the target DBMS are derived from the user presented data structures. Database terminals, in contrast, provide the user with a constant data model and DML across heterogeneous DBMSs. The target data structures are fixed and

the data structures presented to the user are derived from these target data structures [KLUGA 78].)

In both of these cases only one set of data structures is fixed while the other is derived from this fixed set. This allows substantial freedom in tailoring data structures to simplify the translation process. Such freedom is not available in constructing a network data manager in which the data structures presented to the network user are fixed (recall that they were chosen by a committee) and the data structures of the target systems are also fixed.

4.2 XNDM Translation Alternatives

An XNQL statement specifies the sequence of operations to be performed on the underlying information structures. It is a high-level language, and by its very nature, does not specify the step-by-step, system-specific actions needed to evaluate the query by a given target DBMS. It is the function of the translator to supply these details.

Since XNQL is a query language, the primitive information structures of the language are aggregated, not simple, data. That is, the basic 'atoms' of data expressed in an XNQL statement are relations rather than individual data elements. The translator interprets these data objects in terms of the primitive data constructs provided by the particular target DBMS and its data structuring rules.

Construction of the XNQL translator is further complicated by the fact that different target systems support different primitive operations and data structures; therefore we need not a single translator but a family of translators. Two approaches to their realization can be identified: construction of a collection of source-target specific translators or, alternatively, construction of a single translator for the bulk of the translation process common to all translators together with custom tailored front ends handling the source specific portion of the translation process and custom tailored back ends handling the target specific portion of the translation process.

Construction of independent translators has the advantage that design unity and run-time efficiency is more achievable with a single translator for each target DBMS. However, an entire translator is needed to support each additional target, whereas in the family approach all the translators share a core design which defines the common (source and target-independent) part of the translator. Each new translator in the family is obtained by building source and target-oriented specialities on top of the basic design. Therefore the bulk of the implementation effort is available across different target systems and new developments need not start from scratch.

An important side-effect of the family approach is the insight it provides for DBMS data manipulation and structuring facilities. That is, a simple, coherent design for a translator family is impossible without abstracting the essential properties of target systems and recognizing their commonalities and differences. Thus, we have chosen the approach of designing a good general framework, i.e. a consistent, efficiently implementable translator allowing effective use of target system facilities. The insights provided by this framework are augmented by those developed in preparing the mappings to and from specific target systems.

4.3 XNQL Translation

The complex semantic manipulations required for translation are achieved by means of step-by-step transformations of an appropriately chosen internal representation of the input text. We have chosen a tree as the intermediate representation because of the requirement for flexibility in handling a wide range of target DML's and data structures.

Each transformation takes us somewhat closer to the target query by either changing the original form of the input text to uncover the underlying "basic structure" of the query tree which characterizes the system-independent organization of queries, or reshaping the basic tree to incorporate the surface structure of the target language. The value of this transformational approach is that it reduces the overall translator complexity and also supports a simple, consistent, modular design [DEREF 76].

The translation process is (vertically) segmented into five phases as illustrated in Figure 4-1. A more extensive discussion is contained in [WANGP 79].

Lexical and Syntactic Analysis

The tasks of the lexical and syntactic analysis modules are conventional [GRIED 69]. They produce a source(XNQL)-specific syntax tree representation of the input query. This tree contains all the information originally present in the source text as well as all the information that is inherent in the XNQL grammatical description. The source syntax tree is the first of a sequence of trees used in the translator as intermodular data structures. Each later module takes as input the tree produced by the previous module and leaves a tree that is closer to the target query by reshaping the tree, pruning source-specific information from the tree and/or incorporating target-specific information into the tree. The basic task facing the translator writer is disentangling those aspects of the source and target queries that reflect "essential" (language-independent) logical structures from those that characterize "incidental" (language-specific) representational details.

Standardization

Processing beyond the syntactic level can be made simpler if the source syntax tree is transformed into a standard form where each WHERE clause is represented as a binary tree of predicates connected by AND and OR nodes arranged in conjunctive normal form [STONM 76].

Static Semantic Processing

Since each XNQL query interacts with a data space which is the Cartesian product of several relations subject to the restriction of the WHERE clause, and frequently these restrictions are such that the Cartesian product becomes an equi-join (merging of two relations based on a common column), differences in source and target structures at the record level imply different join conditions in the queries.

The static semantic level of the translator does the processing needed to account for data structure differences at and below the record level by first resolving data item name differences and then the differences in the joins.

The "data item renaming" module traverses the source syntax tree from the top down, replacing all leaf references to source(user) data items with corresponding references to target data items and depositing their attribute information at these nodes. The "record structure mapping" module then deletes all predicate nodes representing joins between different source relations and inserts the appropriate join predicates for target records.

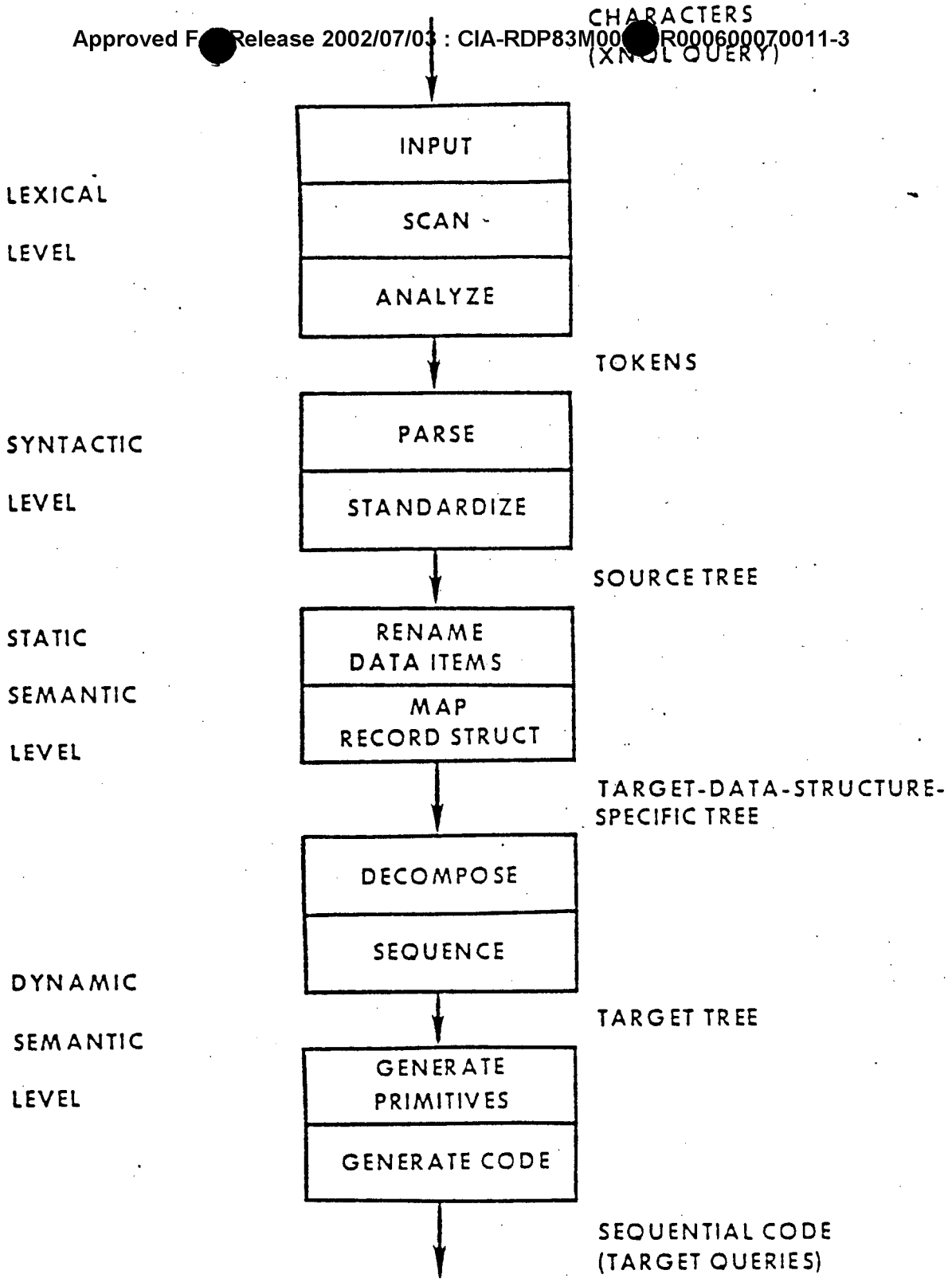


Figure 4-1. The XNQL Translator as a Tree Transformer

Dynamic Semantic Processing

The transformations happening at this level account for the differences in the logical structures of the source and target query languages. Since the unit of data structure for each target query may be smaller than for XNQL (e.g. each Codasyl DML statement can only involve a single record (or set) type, whereas there is no limitation to the number of different tuple types (relations) an XNQL statement can manipulate), we first decompose the query tree into sub-trees, each of which involves a single unit of data structure that a target query can handle. The "sequence" module then chains the sub-trees together in the order that the corresponding queries should be sequenced for the target DBMS and selects the execution sequence of these chains that minimizes the amount of intermediate records needed to be processed.

Code Generation

This is the final phase of the translator and outputs the desired target DML statements that can be executed by the local DBMSs. The first module interprets each of the sub-trees along the chains produced by the Sequencer and generates CALL statements to primitive target database operations. The second (code generation) module then expands these CALLs into sequences of actual target DML statements.

The exact form of the primitives depend upon the particular target system we are considering. Their behavior characteristics fall, in general, into the following categories: search or return the first/next instance of a specified record type, test the truth value of some predicate expression of the record type, partition all instances of a record type on the basis of some data item values and evaluate aggregate functions for the specified record type. (These correspond roughly to the information algebra operations [CODAS 62] of searching/returning the first/next point of a line, bundling, glumping and evaluating functions of lines.)

This extra level of indirection before the actual code generation allows us to separate out the representational details of the target DMLs and makes it possible to have a standard set of primitives for each general class of target systems, that is, Codasyl, relational calculus and relational algebra systems.

The decision to set the primitives at a fairly procedural level (namely, one record instance at a time) was driven by the flexibility it provides for expressing a variety of access strategies. This allows easy incorporation of optimization modules which selects the "best" access paths for the input query based upon knowledge of how the records are stored (keys, inversion indices, etc.). This is particularly important since the value and usefulness of XNDM in a real environment depends critically upon its performance and experiences with current relational DBMSs indicate that some form of optimization is essential in bringing the performance to an acceptable level [SMITJ 75].

5. IMPLEMENTATION STATUS AND CONCLUDING REMARKS

This paper has described the design and ongoing implementation of a collection of functions for providing a uniform network view of data across a heterogeneous collection of network accessible DBMSs. Our experience to date suggests that XNDM is a realistic and pragmatic approach for achieving the advantages of networking given a significant, in place, collection of DBMSs.

Perhaps the three key issues in ensuring user acceptance of a network data manager are: i) access controls and semantic integrity, ii) developing more sophisticated translation capabilities optimizing the allocation of the translation process among NDM and LDBMS, and iii) performance. We believe the basic issues and a reasonable approach for (i) have been discussed in this paper. Developing a more sophisticated translation capability is of obvious importance and closely relates to the performance issue. Implementation of translators should be paralleled with research directed toward a better understanding of the nature of the translation process. Some work is beginning to appear in this area [KLUGA 78] establishing the theoretical limits of translation feasibility.

5.1 Implementation Status

XNDM translation is performed on a PDP-11/45 attached to the Arpanet as are the other host computers. The operating system for the PDP-11/45 is UNIX [THOMK 74] and a the translator is programmed in C. To provide a more uniform interface to the translator, small support modules termed envelopes are implemented on the system on which each LDBMS resides. Basic communications support between systems and the ability to preserve meaning in transporting structured records between heterogeneous systems is provided by an Experimental Network Operating System (XNOS) [KIMBS 78]. Work on the XNQL translator is still in progress. The current version handles two out of the six XNQL constructs (selections of columns and rows), for the following target systems: the Multics Relational Data Store (MRDS) [HONEY 77], a relational calculus system, and the Honeywell 600/6000 Integrated Data Store (IDS) [HONEY 71], a Codasyl-like system. For MRDS, the translator can handle all target data structures in general, but for IDS, target records with multiple owners and multiple members are excluded.

5.2 Implementation Approach

Two different approaches to implementing XNDM can be considered. The first distributes the implementation across the supported host systems while the second, which we have adopted, offloads the implementation, to the extent possible, onto a separate satellite computer.

The tradeoffs between these two approaches are essentially those of evaluating the cost of supporting an additional computer versus the cost of implementing common modules on several different systems. Given the opportunity for centralized design, implementation and support afforded by offloading and the increasingly high cost of software, we believe that offloading is the natural approach in an evolving technology. The alternative might be appropriate for an extremely static environment.

6. ACKNOWLEDGMENTS

The authors would like to express their appreciation to Gary Sockut, Helen Wood, and Fran Nielsen who provided many helpful comments and substantial assistance in preparing this paper.

7. REFERENCES

[ANTHR 65] Anthony, R., "Planning and Control Systems: A Framework for Analysis," Division of Research, Graduate School of Business Administration, Harvard University, 1965.

[ARPAN 76] Arpanet Protocol Handbook, Network Information Center, Stanford Research Institute, Menlo Park, CA, April, 1976.

[BRODM 78] Brodie, Michael, L. "Specification and Verification of Data Base Semantic Integrity", University of Toronto, Computer System Research Group, Technical Report CSRG-91, April, 1978.

[CHAMD 76] Chamberlin, D.D., et al., "SEQUEL 2: A Unified Control", IBM Journal of Research and Development, Nov. 1976, pp. 560-575.

[CODAS 62] Codasyl Development Committee, "An Information Algebra", Comm. of the ACM, Vol. 5, No. 4, April, 1962, pp. 190-204.

[DATEC 77] Date, C.J., An Introduction to Database Systems, Addison-Wesley, Second Edition, 1977.

[DENND 76] Denning, Dorothy E., "A Lattice Model of Secure Information Flow," Comm. of the ACM, Vol. 19, No.5, May, 1976, pp. 236-243.

[DEREF 76] DeRemer, F. L., "Transformational Grammars", in Bauer, F. L., and Eickel, J. (eds) Compiler Techniques - An Advanced Course, Springer-Verlag 1976, pp. 121-145.

[FONGE 79] Fong, Elizabeth, "Semantic Integrity System for an Experimental Network Data Manager", in preparation.

[GRIFP 76] Griffiths, Patricia P. and Bradford W. Wade, "An Authorization Mechanism For a Relational Data Base System", IBM Research RJ 1721, Feb. 1976.

[HONEY 71] Honeywell Information Systems, Inc., Integrated Data Store, Order No. Br69, Rev.1, December, 1971.

[HONEY 77] Honeywell Information Systems, Inc., Multics Relational Data Store (MRDS) Reference Manual, Order No. AW53, Rev.0, September 1977.

[INWG 77] "A Network Independent File Transfer Protocol," prepared by The High Level Protocol Group, INWG Protocol 86, HLP/CP(78)1, December, 1977.

[KARGP 77] Karger, Paul, "Non-Discretionary Access Control for Decentralized Computing Systems," SM Thesis, M.I.T. Dept. of Electrical Engineering and Computer Science, May, 1977. (Also available as MIT/LCS/TR-179, Laboratory for Computer Science,M.I.T., May, 1977, NTIS AD A0::0S08.)

[KIMBS 78] Kimbleton, Stephen R., Helen M. Wood, and M. L. Fitzgerald, "Network Operating Systems - An Implementation Approach", Proc. National Computer Conference, AFIPS Press, Anaheim, CA, Vol. 47, June, 1978, pp.773-782.

[KLUGA 78] Klug, Anthony C. "Theory of Database Mappings", Ph.D. Thesis, Department of Computer Science, University of Toronto, 1978.

[MCLED 76] McLeod, Dennis, "High Level Expression of Semantic Integrity Specifications in a Relational Data Base System", MIT Report MIT/LCS/TR-165, available from DDC AD-A034184.

[NAVAS 76] Navathe, S.B. and J.P. Fry, "Restructuring for Large Databases: Three Levels of Abstraction," ACM Transactions on Database Systems, Vol. 1, No.2, June, 1976, pp. 138-158.

[REISP 75] Reisner, P., R.F. Boyce and D.D. Chamberlin, Human Factors Evaluation of Two Data Base Query Languages: SQUARE and SEQUEL", Proc. National Computer Conference, Anaheim, CA, Vol. 44, May, 1975, pp. 447-452.

[ROTHJ 77] Rothnie, James B. and Nathan Goodman, "An Overview of the Preliminary Design of SDD-1: A System for Distributed Databases", 1977 Berkeley Workshop on Distributed Data Management and Computer Networks, Lawrence Berkeley Laboratory, University of California, Berkeley, CA, May, 1977, pp.39-57. (Also available from Computer Corporation of America, 575 Technology Square, Cambridge, MA 02139, as Technical Report No. CCA-77-04).

[SMITJ 75] Smith, John Miles and Philip Yen-Teng Chang, "Optimizing the Performance of a Relational Algebra Database Interface", Comm. of the ACM, Vol. 18, No. 10, October 1975, pp. 568-579.

[STONM 76] Stonebraker, M., Eugene Wong, Peter Krepts, and Gerald Held, "The Design and Implementation of INGRES", ACM Transactions on Database Systems, Vol. 1, No. 3, September, 1976, pp. 189-222.

[STONM 77] Stonebraker, M., and E. Neuhold, "A Distributed Database Version of INGRES", 1977 Berkeley Workshop on Distributed Data Management and Computer Networks, Lawrence Berkeley Laboratory, University of California, Berkeley California, May 1977, pp. 19-36.

[THOMK 74] Thompson, K. and D. Ritchie, "The UNIX Time-Sharing System," Comm. of the ACM, Vol. 17, No. 7, July, 1974, pp. 365-375.

[WANGP 79] Wang, Pearl S.-C., "Common Query Language for the Networking Environment: Design, Translation Structure, and Initial Implementation," in preparation.

[WOODH 79] Wood, Helen M. and Stephen R. Kimbleton, "Access Control Mechanisms for a Network Operating System," to appear, Proc. National Computer Conference, June, 1979.